COIT11134 Object-Oriented Programming
Practical Assignment 2

Due date:	**Week 10 - Friday 11:59 pm AEST**
Weighting:	30%
Length:	N/A

Objectives

This assessment item relates to the following unit learning outcomes:

- Implement and test object-oriented programs using a modern programming language,
- Build interactive software applications using Graphical User Interface components, and
- Apply the concept of exception handling and file data manipulation in object-oriented code.

Overview

For this assignment, you are required to develop a **Java GUI Application (JavaFX with NetBeans Maven)** to demonstrate that you can use Java constructs including input/output via the GUI, Java data types, Java classes and objects, Java ArrayList, selection and looping statements and various other Java features including file handling.

Some of the JavaFX GUIs are provided in this document. The respective FXML skeleton files and partial solutions will be available on the unit websites. However, you are encouraged to design your GUIs on your own instead of using the supplied skeleton GUIs. You can adopt any scene-layout patterns that are suitable for the requirements of this assignment. You will be using the topics learnt from Weeks 1 to 9.

Apart from developing the program, **a report** showing the screenshots of the output produced including annotations of your testing must be submitted.

Assignment Specification

Assignment 2 is an extension of Assignment 1, plus some extra functionalities are required.

CQU Community College is satisfied with the prototype of menu-driven Java console application that you have developed for accommodation registration. They have requested you develop a Java GUI application for data entry and manipulation.

You can modify and reuse the code from your Assignment 1 and integrate them into a GUI application.

Main Functions

1. The user should be able to add new accommodations via a GUI.
2. The user should be able to search for a specific accommodation (new function).
3. The user should be able to view all the accommodations.
4. The user should be able to read the existing records from a file (new function).
5. The user should be able to save the updated accommodation list to the file (new function).

Graphical User Interface Samples

(Note: You don't need to follow these scenes. You can design your scenes as many as you like).

You will need to create three FXML files for three scenes that will be displayed in this program. The specifications for the scenes are as follows.

1. Main menu scene

The main menu scene is illustrated in Figure 1 below. The main menu scene controller will switch to the appropriate scenes based on the user's choice. In the supplied skeleton, the *mainMenu.fxml* file contains this scene and *MainMenuController.java* contains the associated controller code for the main menu scene. The "Add new record" and "Search for a record" buttons will switch the scene to "Add new record scene" (Figure 2) and "Search for record scene" (Figure 3) respectively. The "View all records" button will display all the accommodations in the text area in the centre of the main menu scene. The "Save and Exit" button will terminate the program and save all the current data (all the accommodations in the ArrayList) to a text file. The "View all record" and "Save and Exit" do not need separate scenes.

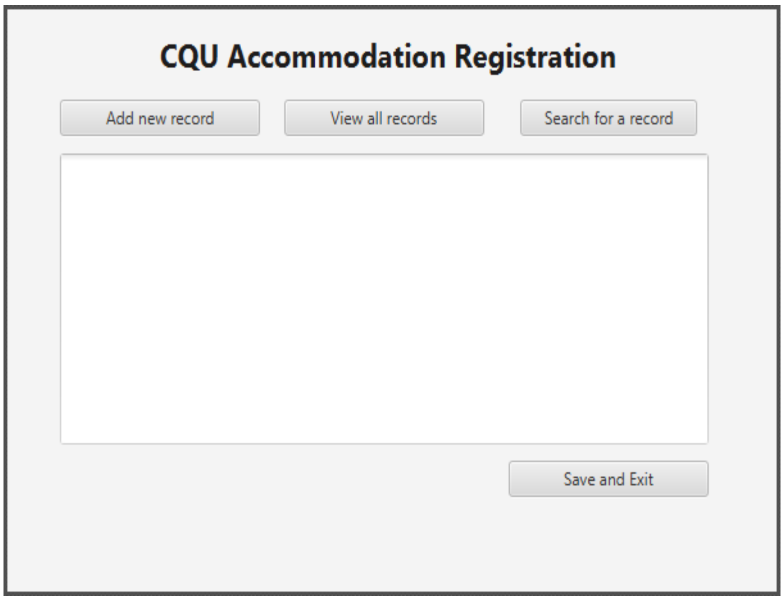

Figure 1 – Main menu scene

2. Add new record scene

This scene contains seven input fields to take an accommodation entry and three buttons, as illustrated in Figure 2. The scene controller will allow the user to enter an accommodation record, create an accommodation object (via the "Submit" button) and append the object to the ArrayList. The scene controller will display a message if the new record has been saved successfully, clear all the fields and return focus back to the first TextField (Accommodation ID). In the provided code skeleton, the *addRecord.fxml* and *AddRecordController.java* have this scene and associated code. The "Reset" button will erase all the text in all text fields.

The “Cancel” button will direct the user to the Main menu scene. You will need to complete the controller code for the required functionalities.

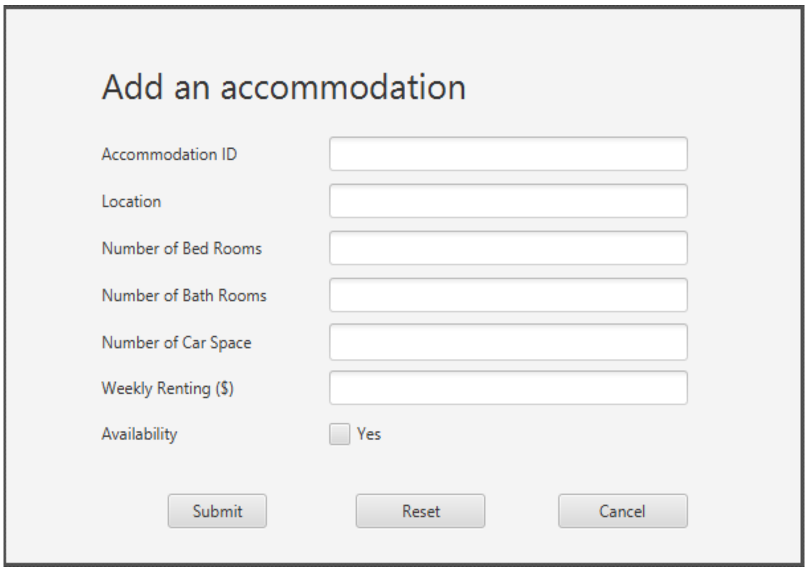

Figure 2 - Add new record scene

3. Search record scene

The search record scene is illustrated in Figure 3. This scene will allow the user to enter an accommodation ID and press the “Search” button to search for a specific accommodation. The search result will be displayed in the text area. The “Exit Search” button will take the user back to the main menu scene. The *searchRecord.fxml* and *SearchRecordController.java* have this skeleton code. You will need to implement the associated code to perform the search function.

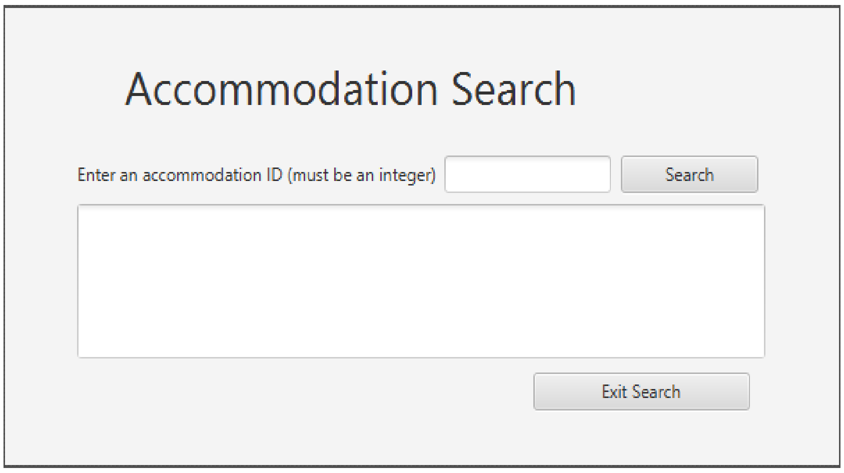

Figure 3 – Search record scene

Data Validation and Data Integrity

Your program should validate the user's inputs. You can implement this after you have got the basic functionality implemented. The validation requirements are the same as the Assignment 1, as below,

- Accommodation ID cannot be duplicated
- Accommodation ID must be an integer
- The number of bedrooms, bathrooms and car spaces must be numeric values and cannot be negative
- Weekly rent must be a numeric value and cannot be negative
- Availability input should be a Boolean data type.

Save and Load Records from Files

All the accommodations need to be stored in a text file. You need to implement file reading and writing for accommodation objects. **This is a new function required** in Assignment 2.

When the application is started, it should get/load the data from the relevant file related to accommodations that have been already entered earlier; it then stores the data in an ArrayList. If there are no accommodations so far, display an appropriate message in an appropriate way (such as a pop-up message box). As the application runs, the new records will be appended to the ArrayList. During application execution, the ArrayList will contain all objects created so far across all sessions.

When "Save and Exit" from the main menu, the application will save all the objects from the ArrayList as records in the relevant text file.

Supplied Code

A draft code skeleton GUI will be provided on the unit website. Some functions are partially provided. You have to investigate the code, complete the remaining scenes and fully develop all associated controller code.

(Note: You don't have to use the skeleton GUI provided. You are encouraged to design and implement your GUI and controllers).

Please check the marking criteria to ensure you have completed all the requirements.

Assignment submission

You must submit your assignment using the Moodle online submission system.

You should submit two files:

1. A zipped (by export project via NetBeans) project file containing all the folders of your project.
2. A Word file containing **a report** showing the screenshots of the output produced including annotations of your testing.

Assignment 2 Marking criteria

Item	Description	Maximum
1	Runs without any compilation/run-time errors.	1
2	Reads and stores the input data from the relevant file into a suitable data structure	1
3	Displays error message when the input file is empty (1). If it is not empty, the system displays the latest record details initially (1)	2
4	Main menu scene: • Add a new record – switch to the associate scene (1) • Search for a record – switch to the associate scene (1) • View all records – display all the current records (4) • Save and exit – Saves/appends the data into the relevant files (2)	8
5	Add record scene: • Input data validations (2) • Submit function – an object is created and added to the list (4) • Reset function – erase all input fields (1) • Cancel function – go back to main menu (1)	8
6	Search record scene: • Input data validation (1) • Search function (3) • Exit Search (1)	5
7	Uses Exception handling and provides appropriate error messages	1
8	Uses good programming practice (comments, indentation, naming and readability)	2
9	Report doc - Word file contains assumptions (if any), test plans and Screenshots	2
	Total	30

Note: *If the program does not compile/run, partial marks may be given appropriately based on the source code inspection.*